LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-CONF-208379

AN INCOMPRESSIBLE ALE METHOD FOR FLUID-STRUCTURE INTERACTION

Timothy A. Dunn

December 3, 2004

NECDC
Livermore, CA, United States
October 4, 2004 through October 7, 2004

An Incompressible ALE Method for Fluid-Structure Interaction

Timothy A. Dunn

Lawrence Livermore National Laboratory

An ALE finite element method was developed to investigate fluid-structure interaction. The write-up contains information about the method, the problem formulation, and some results from example test problems.

1 Introduction

Multi-disciplinary analysis is becoming more and more important to tackle todays complex engineering problems. Therefore, computational tools must be able to handle the complex multi-physics requirements of these problems. A computer code may need to handle the physics associated with fluid dynamics, structural mechanics, heat transfer, chemistry, electro-magnetics, or a variety of other disciplines–all coupled in a highly non-linear system. The objective of this project was to couple an incompressible fluid dynamics package to a solid mechanics code. The code uses finite-element methods and is useful for three-dimensional transient problems with fluid-structure interaction. The code is designed for efficient performance on large multi-processor machines.

2 Fluid Methodology

The main code-development effort associated with this project involved modification to the incompressible flow package. The code was originally developed in an Eulerian reference frame. To couple the fluid to the solid motion, the flow solver needed to handle moving geometries. Therefore, the fluid equations were reformulated to account for grid motion using an ALE formulation.

2.1 The Arbitrary Lagrangian/Eulerian Method

Several numerical methods are available for fluid flow problems. Based on the arrangement of the computational mesh, these methods can be grouped into the

Lagrange method, the Euler method, or the arbitrary Lagrangian Eulerian (ALE) method. Each of these methods has advantages and disadvantages in solving flow problems with moving boundaries.

In the Lagrangian description, the mesh moves with the fluid motion. Thus, moving boundaries are easily expressed. This is the technique typically used in structural mechanics codes. However, for fluid problems, the computation can easily encounter grid tangling caused by large boundary motions or complex motion of the flow-field.

The Eulerian method employs a mesh which is fixed in space. Here the fluid is allowed to flow through the mesh. This is the usual technique used for fluids problems. Grid tangling is not an issue for a fixed grid and complex fluid motions can be studied. However, without special treatment, the conservation equations are not exactly satisfied at moving boundaries.

As its name implies, the ALE method is a hybrid of the Lagrangian and Eulerian methods. In the ALE method, the mesh is arranged independent of the fluid motion. The grid can be moved to follow boundary motion, resolve complex flow features, and prevent the grid from tangling. Due to its general applicability, the ALE method was chosen for use in our current code.

From a mathematical point of view, the three methods differ in the reference frame in which the derivatives are expressed. This boils down to how the material derivative is evaluated. A nice discussion of each of these reference frames is presented by Uchiyama [1]. The transport of a generic variable (ϕ) in one-dimension will be used to help describe the difference in the three approaches. In the following equations, X denotes the material coordinate system, x denotes the spatial coordinate system, and χ denotes the referential coordinate system:

Lagrangian: Material Reference Frame

$$\left.\frac{\partial \phi(X,t)}{\partial t}\right|_X = f \tag{1}$$

Here the derivatives express the change in the transport variable as it moves with the material.

Eulerian: Spatial Reference Frame

$$\left.\frac{\partial \phi(x,t)}{\partial t}\right|_x + u\frac{\partial \phi}{\partial x} = f \tag{2}$$

Here the derivatives express the change in the transport variable at a point fixed in space. The advective derivative accounts for the material flowing through the fixed

reference frame where the material velocity is given as

$$u = \left.\frac{\partial x}{\partial t}\right|_X$$

The advective term is non-linear and therefore requires special consideration when solving.

ALE: Referential Reference Frame

$$\left.\frac{\partial \phi(\chi, t)}{\partial t}\right|_\chi + (u - \hat{u})\frac{\partial \phi}{\partial x} = f \tag{3}$$

Here the derivatives express the change in the transport variable at a point moving with the reference frame. The advective derivative not only accounts for the motion of the material, but also the motion of the reference frame, given by

$$\hat{u} = \left.\frac{\partial x}{\partial t}\right|_\chi$$

Eqn. 3 can be solved directly or the convective term can be split out as a second step.

$$\begin{aligned} \left.\frac{\partial \phi(x, t)}{\partial t}\right|_x + u\frac{\partial \phi}{\partial x} &= f \\ \left.\frac{\partial \phi(\chi, t)}{\partial t}\right|_\chi + (-\hat{u})\frac{\partial \phi}{\partial x} &= 0 \end{aligned} \tag{4}$$

This method is common in hydrocodes as discussed in Benson [2]. Both methods have been implemented for this study. Part of this project is to compare the results obtained from each method. Eqn. 3 will be referred to as Advection Option #1 and Eqn. 4 will be called Advection Option #2.

2.2 The Incompressible Flow Equations

For the purposes of this paper, an incompressible fluid will be defined as one where the density is constant in both space and time. The motion of an incompressible viscous fluid is governed by a set of partial differential equations that arises from the laws of conservation for a physical system. The physical laws of interest are the conservation of mass (continuity) and the conservation of momentum (Newton's second law). Together, these coupled equations are known as the Incompressible Navier-Stokes equations. Their derivation can be found in any standard text on fluid mechanics. The mass and momentum equations can be

expressed respectively in differential form as

$$\begin{aligned} \frac{\partial u_\beta}{\partial x_\beta} &= 0 \\ \frac{\partial u_\alpha}{\partial t} + u^*_\beta \frac{\partial u_\alpha}{\partial x_\beta} &= \frac{\partial \tau_{\alpha\beta}}{\partial x_\beta} + g_\alpha \end{aligned} \tag{5}$$

where the Greek subscripts α and β represent the three spatial coordinate directions, and repeated indices imply summation. The pseudo-stress term is defined as

$$\tau_{\alpha\beta} = -P\delta_{\alpha\beta} + (\nu + \nu_t)\frac{\partial u_\alpha}{\partial x_\beta} + \nu_t \frac{\partial u_\beta}{\partial x_\alpha} \tag{6}$$

and the rest of the symbols are defined in the table at the end of the section. The eddy-viscosity coefficient (ν_t) arises from an averaging or filtering technique and allows for the inclusion of a turbulence model. The eddy viscosity equals zero if no turbulence model is used.

These equations are usually formulated in an Eulerian reference frame. Here, the equations are presented in the ALE formulation where the advective velocity (u^*) accounts for the grid motion as described in Eqn. 3. The Incompressible Navier-Stokes equations are complex, non-linear, and difficult to solve. Thus, numerical techniques are required for all but the simplest problems.

Notation:

g_α	=	body force (acceleration due to gravity, etc.)
ν	=	kinematic viscosity
ν_t	=	kinematic eddy viscosity
P	=	kinematic pressure = $\frac{p}{\rho}$
p	=	static pressure
ρ	=	fluid density
$\delta_{\alpha\beta}$	=	Kronecker delta
t	=	time
u_α	=	fluid velocity component
u^*	=	advective velocity = $(u - \hat{u})$
$\hat{u}$	=	velocity of the reference frame (grid)
x_α	=	coordinate component

2.3 The Weak Form

Let us cast Eqn. 5 in a general form, represented by operator A, acting on domain Ω, such that

$$A(u, P) = 0 \tag{7}$$

where u and P are the solution. Since u and P are not known, define discrete functions $\bar{u}$ and $\bar{P}$ to approximate the solution functions. In general, the

approximate solutions will not satisfy the original equation on a point-by-point basis, resulting in some local error

$$\epsilon = A(\bar{u}, \bar{P}) \neq 0 \tag{8}$$

at every point in the domain. This error can be minimized with the method of weighted residuals (MWR). Apply the MWR to Eqn. 8 by multiplying by a spatially varying weighting function $v(\Omega)$ and integrating over the domain

$$\int_\Omega v(\Omega)\epsilon \, d\Omega = \int_\Omega v(\Omega) A(\bar{u}, \bar{P}) \, d\Omega = 0 \tag{9}$$

in order to satisfy the original equation in an average sense over the domain.

To apply the method of weighted residuals to the system of equations in Eqn. 5, multiply each equation by a test function. Use w for the mass equation and v for the momentum equation. Next integrate over Ω

$$\begin{gathered} \int w \frac{\partial \bar{u}_\beta}{\partial x_\beta} = 0 \\ \int v \frac{\partial \bar{u}_\alpha}{\partial t} + \int v \bar{u}^*_\beta \frac{\partial \bar{u}_\alpha}{\partial x_\beta} - \int v \frac{\partial \bar{\tau}_{\alpha\beta}}{\partial x_\beta} - \int v g_\alpha = 0 \end{gathered} \tag{10}$$

where $d\Omega$ is implied inside the integral. The stress term can be simplified, i.e. the order of the derivatives can be reduced. First integrate by parts

$$\int v \frac{\partial \bar{\tau}_{\alpha\beta}}{\partial x_\beta} = -\int \bar{\tau}_{\alpha\beta} \frac{\partial v}{\partial x_\beta} + \int \frac{\partial}{\partial x_\beta} \left(v \bar{\tau}_{\alpha\beta} \right)$$

Next, the last term in the above equation can be simplified with the divergence theorem

$$\int \frac{\partial}{\partial x_\beta} \left(v \bar{\tau}_{\alpha\beta} \right) = \oint v n_\beta \bar{\tau}_{\alpha\beta}$$

where n_β is the β component of the surface outward unit normal vector, and $\oint$ indicates an integral over the surface boundary, $\partial\Omega$. Combine the above to get the final form of the momentum equation

$$\int v \frac{\partial \bar{u}_\alpha}{\partial t} + \int v \bar{u}^*_\beta \frac{\partial \bar{u}_\alpha}{\partial x_\beta} + \int \bar{\tau}_{\alpha\beta} \frac{\partial v}{\partial x_\beta} - \oint v n_\beta \bar{\tau}_{\alpha\beta} - \int v g_\alpha = 0 \tag{11}$$

Finally, substitute the stress term back in and obtain the final weak form of the mass and momentum equations

$$\begin{gathered} \int w \frac{\partial \bar{u}_\beta}{\partial x_\beta} = 0 \\ \int v \frac{\partial \bar{u}_\alpha}{\partial t} + \int v \bar{u}^*_\beta \frac{\partial \bar{u}_\alpha}{\partial x_\beta} - \int \bar{P} \frac{\partial v}{\partial x_\alpha} + \\ \int \left[(\nu + \nu_t) \frac{\partial \bar{u}_\alpha}{\partial x_\beta} \frac{\partial v}{\partial x_\beta} + \nu_t \frac{\partial \bar{u}_\beta}{\partial x_\alpha} \frac{\partial v}{\partial x_\beta} \right] - \oint v n_\beta \bar{\tau}_{\alpha\beta} - \int v g_\alpha = 0 \end{gathered} \tag{12}$$

2.4 The Galerkin Form

As discussed in Section 2.3, numerical procedures for solving our equations required the replacement of the unknown solution (u, P) with an approximation $(\bar{u}, \bar{P})$ throughout the solution domain Ω. The approximate velocity and pressure will now be defined using the Finite-Element Method. The first step to expand the solution is to discretize the domain Ω, forming a union $\bar{\Omega}$ of elements Ω_e

$$\Omega \approx \bar{\Omega} = \bigcup_e \Omega_e$$

where the sum over e is taken over the total number of elements. We use this discretization to define a set of locally defined basis functions which are pieced together to form a global basis for the approximation subspace. These basis functions will be described in more detail in Section 2.6. At this point we define the approximate solution as a linear combination of the basis functions ϕ and ψ

$$\begin{aligned} u_\alpha(t,x,y,z) \approx \bar{u}_\alpha(t,x,y,z) &= \sum_{j=1}^{N} u_\alpha^j(t)\phi_j(x,y,z) \\ P(t,x,y,z) \approx \bar{P}(t,x,y,z) &= \sum_{j=1}^{M} P^j(t)\psi_j(x,y,z) \end{aligned} \tag{13}$$

where the summations are performed over the total number of velocity nodes N and the total number of pressure nodes M. We can now substitute into the continuity and momentum equations

$$\begin{gathered} \left[\int w\frac{\partial \phi_j}{\partial x_\beta}\right] u_\beta^j = 0 \\ \left[\int v\phi_j\right]\frac{\partial u_\alpha^j}{\partial t} + \left[u_\beta^{*k}\int v\phi_k\frac{\partial \phi_j}{\partial x_\beta}\right]u_\alpha^j - \left[\int \psi_j\frac{\partial v}{\partial x_\alpha}\right]P^j + \\ \left[\int (\nu+\nu_t)\frac{\partial \phi_j}{\partial x_\beta}\frac{\partial v}{\partial x_\beta}\right]u_\alpha^j + \left[\int \nu_t\frac{\partial \phi_j}{\partial x_\alpha}\frac{\partial v}{\partial x_\beta}\right]u_\beta^j - \\ \oint vn_\beta\tau_{\alpha\beta} - \int vg_\alpha = 0 \end{gathered} \tag{14}$$

where summation over j and k are implied and the integrals are now performed over the discretized volume $\bar{\Omega}$. Recall that u^j and P^j are functions of time only and therefore can be pulled out of the volume integrals. The time discretization of these variables will be discussed later.

At this point we will focus on the weighting functions v and w which can be chosen arbitrarily. Notice that the above system contains $3N$ velocity unknowns and M pressure unknowns. In order to solve this system, there must be the same number of equations as unknowns. We can obtain the necessary number of equations by defining one weighting function per unknown. Remember that the

weighting functions were included in the equations to reduce the effects of the error over the area influenced by the weighting functions. Therefore, each weighting function should be chosen to amplify the error over a limited subregion in the vicinity of its corresponding unknown. The most common error distribution principle used for finite elements is the Galerkin method. According to the Galerkin method, the weighting functions are chosen to be the same as the basis functions used to define the approximate solution. Applying this method to our equations, with $w = \psi_i$ and $v = \phi_i$, yields

$$\begin{gathered} \left[\int \psi_i \frac{\partial \phi_j}{\partial x_\beta}\right] u_\beta^j = 0 \\ \left[\int \phi_i \phi_j\right] \frac{\partial u_\alpha^j}{\partial t} + \left[u_\beta^{*k} \int \phi_i \phi_k \frac{\partial \phi_j}{\partial x_\beta}\right] u_\alpha^j - \left[\int \psi_j \frac{\partial \phi_i}{\partial x_\alpha}\right] P^j + \\ \left[\int (\nu + \nu_t) \frac{\partial \phi_j}{\partial x_\beta} \frac{\partial \phi_i}{\partial x_\beta}\right] u_\alpha^j + \left[\int \nu_t \frac{\partial \phi_j}{\partial x_\alpha} \frac{\partial \phi_i}{\partial x_\beta}\right] u_\beta^j - \\ \oint \phi_i n_\beta \tau_{\alpha\beta} - \int \phi_i g_\alpha = 0 \end{gathered} \tag{15}$$

resulting in M continuity equations and N momentum vector equations.

2.5 Matrix Form

The system in Eqn. 15 represents a system of $3N + M$ ordinary differential equations with the same number of unknown time-dependent functions. Each bracketed term, [ ], can be expressed as a matrix, resulting in

$$\begin{gathered} \mathbf{M}\dot{u} + (\mathbf{K} + \mathbf{N}(u^*))\, u + \mathbf{C}P = \mathbf{F} \\ \mathbf{C^T} u = \mathbf{0} \end{gathered} \tag{16}$$

where the matrix entries are computed from integrals of the shape functions. The definitions of each of these matrices are given below.

Mass

$$\mathbf{M} = \begin{bmatrix} m_{ij} & 0 & 0 \\ 0 & m_{ij} & 0 \\ 0 & 0 & m_{ij} \end{bmatrix}, m_{ij} = \int \phi_i \phi_j \tag{17}$$

Diffusion

$$\begin{gathered} \mathbf{K} = \begin{bmatrix} k_{ij_{(xx)}} & k_{ij_{(xy)}} & k_{ij_{(xz)}} \\ k_{ij_{(yx)}} & k_{ij_{(yy)}} & k_{ij_{(yz)}} \\ k_{ij_{(zx)}} & k_{ij_{(zy)}} & k_{ij_{(zz)}} \end{bmatrix}, \\ k_{ij_{(\alpha\beta)}} = \int \left(\delta_{\alpha\beta} (\nu + \nu_t) \frac{\partial \phi_j}{\partial x_\gamma} \frac{\partial \phi_i}{\partial x_\gamma} + \nu_t \frac{\partial \phi_j}{\partial x_\alpha} \frac{\partial \phi_i}{\partial x_\beta}\right) \end{gathered} \tag{18}$$

Advection

$$\mathbf{N}(u^*) = \begin{bmatrix} n_{ij} & 0 & 0 \\ 0 & n_{ij} & 0 \\ 0 & 0 & n_{ij} \end{bmatrix}, n_{ij} = u_\beta^{*k} \int \phi_i \phi_k \frac{\partial \phi_j}{\partial x_\beta} \tag{19}$$

Gradient

$$\mathbf{C} = \begin{bmatrix} c_{ij_{(1)}} \\ c_{ij_{(2)}} \\ c_{ij_{(3)}} \end{bmatrix}, c_{ij_{(\alpha)}} = -\int \psi_j \frac{\partial \phi_i}{\partial x_\alpha} \tag{20}$$

Divergence

$$\mathbf{C^T} = \begin{bmatrix} c_{ji_{(1)}} & c_{ji_{(2)}} & c_{ji_{(3)}} \end{bmatrix}, c_{ji_{(\alpha)}} = -\int \psi_i \frac{\partial \phi_j}{\partial x_\alpha} \tag{21}$$

Force Vector

$$\mathbf{F} = \begin{bmatrix} f_{i_{(1)}} \\ f_{i_{(2)}} \\ f_{i_{(3)}} \end{bmatrix}, f_{i_{(\alpha)}} = \oint \phi_i n_\beta \tau_{\alpha\beta} + \int \phi_i g_\alpha \tag{22}$$

The above equations represent global finite-element matrices. However, their form is identical to the element versions. These integrals can be computed element-by-element and "assembled" to form the global system (see [3]).

2.6 Shape Functions

As discussed in Section 2.4, the approximate solution is expresses as a function of a set of basis functions, a.k.a. shape functions. We can express an arbitrary function, φ, as a linear combination of the shape functions

$$\varphi \approx \sum_{i=1}^{N} N_i \varphi_i \tag{23}$$

where N_i is a globally defined shape function and the coefficients φ_i are the unknown nodal values. It follows that for any node j

$$\varphi_j = \varphi(x_j) = \sum_{i=1}^{N} N_i(x_j) \varphi_i \tag{24}$$

which requires that the shape functions must obey

$$N_i = \begin{cases} 1 & \text{at node } i = j \\ 0 & \text{at node } i \neq j \end{cases} \tag{25}$$

This implies the following identities:

$$\begin{aligned} \sum_{i=1}^{N} N_i &= 1 \\ \sum_{i=1}^{N} N_i x_i &= x \\ \sum_{i=1}^{N} N_i y_i &= y \end{aligned} \tag{26}$$

These relations allow us to define our shape functions locally, element-by-element, in terms of element shape functions $\mathcal{N}$, which are zero outside the element considered. The global functions are obtained by assuming $N_i = \mathcal{N}_i$ within each element.

The Q1Q0 element formulation was chosen for our study. This element is discussed by Gresho, *et.al.* [4]. The velocity shape functions in terms of the element's parametric coordinates (s, t, u) are given as

$$\begin{aligned} \mathcal{N}_1(s,t,u) &= \tfrac{1}{8}(1-s)(1-t)(1-u) \\ \mathcal{N}_2(s,t,u) &= \tfrac{1}{8}(1+s)(1-t)(1-u) \\ \mathcal{N}_3(s,t,u) &= \tfrac{1}{8}(1+s)(1+t)(1-u) \\ \mathcal{N}_4(s,t,u) &= \tfrac{1}{8}(1-s)(1+t)(1-u) \\ \mathcal{N}_5(s,t,u) &= \tfrac{1}{8}(1-s)(1-t)(1+u) \\ \mathcal{N}_6(s,t,u) &= \tfrac{1}{8}(1+s)(1-t)(1+u) \\ \mathcal{N}_7(s,t,u) &= \tfrac{1}{8}(1+s)(1+t)(1+u) \\ \mathcal{N}_8(s,t,u) &= \tfrac{1}{8}(1-s)(1+t)(1+u) \end{aligned} \tag{27}$$

and the pressure shape functions are

$$\mathcal{N}_P(s,t,u) = 1 \tag{28}$$

This element is shown graphically in Figure 1.

2.7 Parametric Coordinate Transformations

The element shape functions given in Section 2.6 are given in the element's parametric coordinate system. However, the integrals making up the system matrices must be evaluated using the global coordinate system. Therefore, a series of transformations must be performed to evaluate the integrals. The global

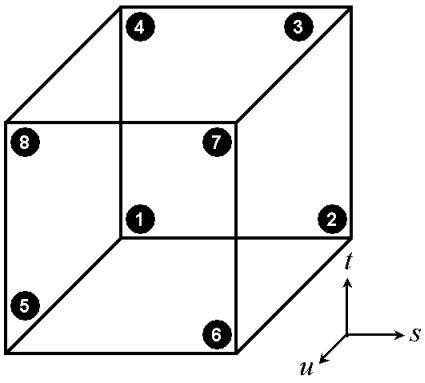

Figure 1: Node numbering of element in local coordinates.

coordinates can be expressed in terms of the parametric coordinates as

$$\begin{aligned} x(s,t,u) &= \sum_{i=1}^{n_e} \mathcal{N}_i(s,t,u)x_i \\ y(s,t,u) &= \sum_{i=1}^{n_e} \mathcal{N}_i(s,t,u)y_i \\ z(s,t,u) &= \sum_{i=1}^{n_e} \mathcal{N}_i(s,t,u)z_i \end{aligned} \tag{29}$$

where

$$\mathcal{N}_i(s,t,u) = N_i\left[x(s,t,u), y(s,t,u), z(s,t,u)\right]$$

Therefore, the derivatives of the shape functions can be expressed using the chain rule as

$$\begin{aligned} \frac{\partial \mathcal{N}_i}{\partial s} &= \frac{\partial N_i}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial s} + \frac{\partial N_i}{\partial z}\frac{\partial z}{\partial s} \\ \frac{\partial \mathcal{N}_i}{\partial t} &= \frac{\partial N_i}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial N_i}{\partial z}\frac{\partial z}{\partial t} \\ \frac{\partial \mathcal{N}_i}{\partial u} &= \frac{\partial N_i}{\partial x}\frac{\partial x}{\partial u} + \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial u} + \frac{\partial N_i}{\partial z}\frac{\partial z}{\partial u} \end{aligned} \tag{30}$$

or equivalently in matrix form as

$$\begin{pmatrix} \frac{\partial \mathcal{N}_i}{\partial s} \\ \frac{\partial \mathcal{N}_i}{\partial t} \\ \frac{\partial \mathcal{N}_i}{\partial u} \end{pmatrix} = \begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} & \frac{\partial z}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} & \frac{\partial z}{\partial t} \\ \frac{\partial x}{\partial u} & \frac{\partial y}{\partial u} & \frac{\partial z}{\partial u} \end{bmatrix} \begin{pmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{pmatrix} = J \begin{pmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{pmatrix} \tag{31}$$

where J is the Jacobian matrix. The Jacobian matrix can be expressed as derivatives of the element shape functions

$$J_{\alpha\beta} = \frac{\partial x_\alpha}{\partial s_\beta} = \sum_{i=1}^{n_e} \frac{\partial \mathcal{N}_i(s,t,u)}{\partial s_\beta} x_{\alpha,i} \tag{32}$$

The shape function derivatives in terms of the global coordinate system can be computed using the inverse of the Jacobian matrix

$$\begin{pmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{pmatrix} = J^{-1} \begin{pmatrix} \frac{\partial \mathcal{N}_i}{\partial s} \\ \frac{\partial \mathcal{N}_i}{\partial t} \\ \frac{\partial \mathcal{N}_i}{\partial u} \end{pmatrix} \tag{33}$$

These values can then be used to evaluate the integrals in the matrix equations. From Calculus we know that the determinant of the Jacobian matrix can be used to transform an integral from one coordinate system to another using the relation

$$d\Omega = dx\, dy\, dz = detJ\, ds\, dt\, du$$

Therefore, the matrix integrals can be evaluated with the transformation

$$\begin{aligned} \int_{\Omega^e} \varphi\, d\Omega &= \int_{\Omega^e} \varphi(x,y,z)\, dx\, dy\, dz \\ &= \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \varphi(s,t,u)\, detJ\, ds\, dt\, du \end{aligned} \tag{34}$$

These integrals are then evaluated using Gaussian quadrature as described by Comini [3]. Typically eight quadrature points are used. Single-point integration is also useful, but requires hour-glass stabilization as discussed in Goudreau and Hallquist [5].

2.8 Time-stepping Algorithm

Eqn. 16 represents a system of ordinary differential equations which must be solved in time. This presents a challenge, as this system is very difficult to solve efficiently in its coupled form. Therefore, we use a segregated solve to obtain the velocity and pressure solutions in separate steps. In particular, we use an implicit projection method to discretize the ODEs. This procedure is described in Gresho, *et.al.* [6]. The philosophy behind projection methods is to provide a way to decouple the pressure and velocity fields to provide an efficient computational method to solve transient incompressible flow problems.

First the momentum equation is discretized using a two-level split-theta time-stepping scheme

$$\begin{gathered} \left[\mathbf{M} + \Delta t\theta_K \mathbf{K} + \Delta t\theta_N \mathbf{N}(\tilde{u}^*)\right] \frac{\tilde{u}}{\Delta t} = \\ \mathbf{M}\frac{u^n}{\Delta t} - (1-\theta_K)\,\mathbf{K}u^n - (1-\theta_N)\,\mathbf{N}(u^{*n})u^n + \mathbf{F}^n - \mathbf{M}\mathbf{M_L}^{-1}\mathbf{C}P^n \end{gathered} \tag{35}$$

This equation is solved for an intermediate velocity $\tilde{u}$ which does not satisfy the divergence-free condition in the continuity equation. The value of the θs can be chosen to tailor your time-stepping scheme. Use $\theta = 0$ for an explicit forward-Euler scheme, $\theta = 1$ for implicit backward-Euler, or $\theta = 1/2$ for Crank-Nicholson. The

right-hand-side of this equation is computed using the pressure from the previous time-step and the non-linear advection term is computed using a linearized guess for the new velocity

$$\tilde{u}^* = u^n + (\Delta t)\frac{\partial u^n}{\partial t} + \cdots$$

For all results presented here, a zeroth-order approximation is used such that

$$\tilde{u}^* = u^n$$

After we implicitly move the material with the momentum equation, the discrete pressure-Poisson equation (PPE) is solved to enforce the continuity equation.

$$\begin{aligned}\left[\mathbf{C^T}{\mathbf{M_L}}^{-1}\mathbf{C} - \mathbf{S}\right]\lambda &= \mathbf{C^T}\frac{\tilde{u}}{\Delta t} \\ &= \mathbf{C^T}{\mathbf{M_L}}^{-1}\mathbf{M_L}\frac{\tilde{u}}{\Delta t}\end{aligned} \tag{36}$$

where λ is a Lagrange multiplier used to update the pressure. $\mathbf{M_L}$ is the lumped mass matrix and $\mathbf{S}$ is a stabilization matrix used to remove spurious pressure modes [7]. Note that multiplying the lumped mass matrix by its inverse on the right-hand-side is done purely for computational reasons to enforce boundary conditions. We end the step by computing the final values of the pressure and velocity

$$\begin{aligned}\lambda = P^{n+1} - P^n \Rightarrow P^{n+1} &= \lambda + P^n \\ u^{n+1} = \Delta t{\mathbf{M_L}}^{-1}&\left(\mathbf{M_L}\frac{\tilde{u}}{\Delta t} - \mathbf{C}\lambda\right)\end{aligned} \tag{37}$$

3 Solid Methodology

The implicit structural mechanics algorithm for solid materials is based on an updated Lagrangian formulation. Dynamic equilibrium is obtained by solving a set of non-linear equations using the state and configuration at the end of the time-step. Non-linearities arise due to material response and configuration changes. The method handles these non-linearities using a Newton-Raphson iteration to obtain the final configuration.

The local dynamic equilibrium relation is given by

$$\rho\ddot{u}_\alpha = \frac{\partial \sigma_{\alpha\beta}}{\partial x_\beta} + \rho b_\alpha \tag{38}$$

where σ is the stress tensor, b is a body force vector per unit mass, ρ is the density, and $\ddot{u}$ is the second time derivative of the nodal displacement vector. The principle of virtual work is used to obtain a weak form of the equation and the finite-element method is used to discretize and solve the equation. This analysis can be found in the report by Becker [8] and will not be repeated here as its derivation was not the focus of the current project.

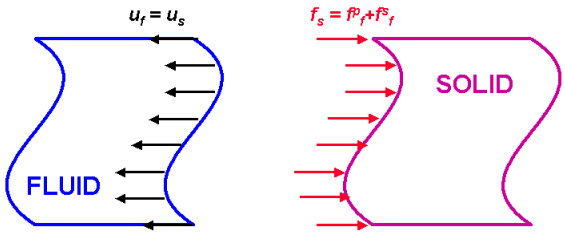

Figure 2: Fluid and solid are coupled through the boundary conditions.

4 Coupling the Fluid & Solid Codes

The objective of this project was to couple an incompressible fluid dynamics package to an implicit structural mechanics code. The idea was to preserve the capabilities of each individual package while allowing them to interact. Due to the transient nature of the solution procedure, a loosely-coupled approach can be employed. This is sometimes referred to as an operator-split approach. In this method each set of equations is solved separately at each time-step. It is assumed that the non-linearity between the fluid and solid system is minor for a single time-step and therefore no iteration between the fluid and solid solve is performed.

The fluid and solid material only interact at their interface. Thus, the fluid-structure coupling occurs through the boundary conditions. This is seen in Figure 2 which shows the interface between the fluid and the solid material. The interface between the fluid and solid will be called $\partial\Omega_{int}$. Currently the nodal positions of the fluid and solid interface nodes must coincide, i.e. the fluid and solid share nodes at the interface. At the interface the fluid material applies a force boundary condition to the solid

$$\mathbf{n} \cdot \sigma = \mathbf{t}_{fluid} \qquad \text{on } \partial\Omega_{int}$$

where $\mathbf{t}_{fluid}$ is the applied surface traction force per unit area. This is a Neumann (Natural) boundary condition on the solid. The traction is computed and integrated during the fluid time-step using the pressure and shear forces within the fluid. This applied force will prescribe the motion of the solid system. The velocity of the solid at the fluid-structure interface will then be used as an essential (Dirichlet) boundary condition on the fluid

$$\mathbf{u} = \mathbf{V}_{solid} \qquad \text{on } \partial\Omega_{int}$$

where $\mathbf{u}$ is the fluid velocity vector and $\mathbf{V}_{solid}$ is the calculated solid velocity at the interface.

The steps involved with each time-cycle are outlined below:

1. Solid Solution
    - Solve for new solid configuration (using force obtained during previous fluid cycle)
    - Solve for velocity at the interface
2. Mesh Relaxation
    - Move fluid grid to account for new solid configuration
    - Compute grid velocity based on change in grid position
    - Advect fluid variables (Advection Option #2 Only)
3. Fluid Solution
    - Solve for fluid velocity and pressure
    - Compute force acting on interface
4. Complete the Cycle
    - Post-Process
    - Increment Time
    - Return To Step 1

5 Results

Two test cases were run for this project. The first test looked at flow around a rigid sphere with a moving grid. This test was designed to validate the grid motion capability of the incompressible flow model. Therefore, the solid mechanics code was not utilized for this test. The second test also looked at flow around a moving sphere, but the sphere was solid and it's motion was computed by the solid mechanics code.

5.1 Test Case #1: Flow Around a Sphere

Test problem #1 looked at flow around a three-dimensional circular sphere. In this case the sphere was rigid and the flow conditions were

$D = 1$	Diameter of Sphere
$U_{\text{inf}} = 1$	Freestream Velocity
$\nu = 0.1$	Kinematic Viscosity
$\Rightarrow Re = 10$	Reynolds Number

At a Reynolds Number of 10, the flow has not yet separated but the streamlines are asymmetric (see Van Dyke [9]). In order to test the grid motion, this test was run on a fixed grid with a uniform freestream velocity and on a moving grid where the

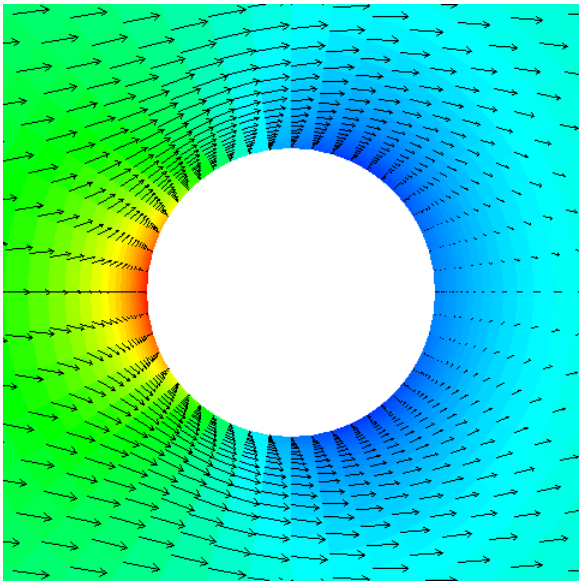

Figure 3: Color contours of pressure overlaid with velocity vectors for fixed grid.

freestream fluid was at rest and the sphere was translated. Both advection options were tested using the moving grid. If the grid motion is working correctly, all three cases should result in the same answer.

Four levels of grid resolution were used for each case to study convergence. The number of elements for each grid were 1152, 9216, 31104, and 73728. This corresponds to the coarsest grid times 1^3 ,2^3, 3^3, and 4^3, where the power of 3 is due to refinement in the 3 spatial directions. These grids will be referred to as grids 1, 2, 3, and 4 respectively. All cases ran for 2500 cycles with an initial time-step of 1E-10, which grew exponentially to 1E-2.

These are three-dimensional transient simulation. As such, they required a fairly large amount of computational resources. Grid 4 was run on 24 (fast) Linux processors. It required a little over 4 hours of wall-clock time to perform the 2500 steps. The smaller grids required substantially smaller compute times.

We first look at the fixed grid simulation. Figure 3 shows the final flow-field in the simulation for grid 3. Velocity vectors are shown on top of pressure contours. As can be seen in the figure, a large boundary layer has developed around the sphere. A large stagnant flow region is present behind the sphere, but no separation has occurred. This agrees well with the qualitative data found in Van Dyke [9].

Figure 4 plots the time-history of the drag acting on the sphere. For all grids the drag starts out at a very large value then gets smaller and converges to a "steady-state" value. Figure 5 plots the final drag (cycle 2500) for each grid to see the spatial grid convergence. The plot shows that the change in drag is reduced at every level of grid refinement indicating grid convergence.

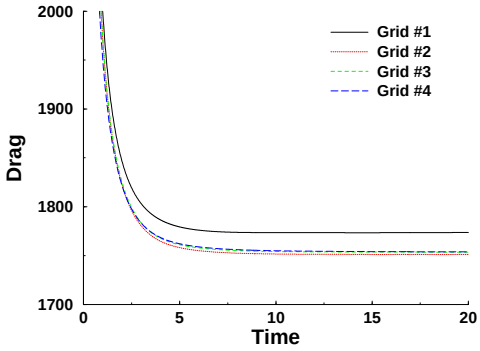

Figure 4: Time-history plot of drag for fixed grid simulation.

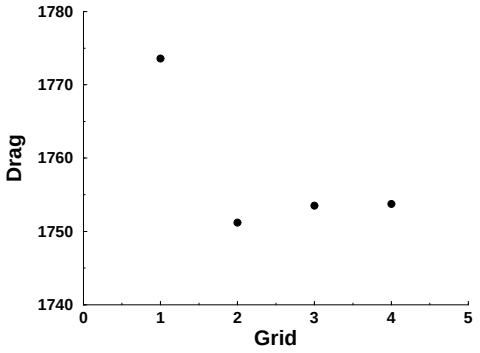

Figure 5: Convergence plot of drag for fixed grid simulation.

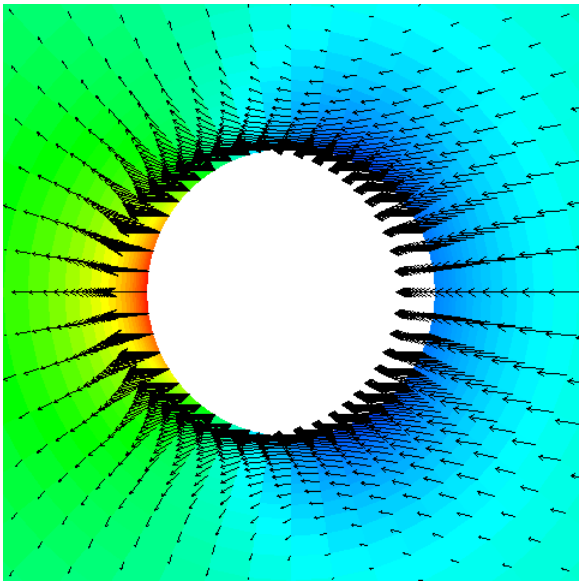

Figure 6: Color contours of pressure overlaid with velocity vectors for advection option #1.

Next we look at the moving grid simulation using Advection Option #1 (i.e. Eqn. 3). Figure 6 shows the final flow-field in the simulation for grid 3. Velocity vectors are shown on top of pressure contours. Figure 7 shows the same plot, but the velocity vectors are relative to the sphere motion ($u - \hat{u}$). This lets us see that Option #1 gives similar qualitative results as the fixed grid solution.

Figure 8 plots the time-history of the drag and Figure 9 shows the convergence history of the drag. Here we see some strange behavior in the results for grid 4. It is assumed that this was caused by a time-step that was too large for this grid refinement. A drawback to the projection algorithm used in the flow code is that although it is implicit and is not constrained by a Courant time-step limit, time-steps many times larger than the Courant limit can still cause instabilities due to poor linearization of the advective operator. Many time-steps were experimented with, and although it did look like smaller time-steps would produce better results, computational expense prohibited their use. More study is still required to fully understand this phenomena.

Now we look at the moving grid simulation using Advection Option #2 (i.e. Eqn. 4). Figure 10 shows the final flow-field in the simulation for grid 3 with velocity vectors on top of pressure contours. Figure 11 shows the same plot, but with relative velocity vectors. These figures show that this method produces a very different result than Option #1 or the fixed grid simulation. This is not unexpected. These simulations were performed with a time-step that was *way* too big for this method. Option #2 uses an explicit scheme to advect the flow quantities, but the time-step used was much larger than the Courant limit. Although the code did not crash due to instabilities, the values coming out of the advection routines cannot be

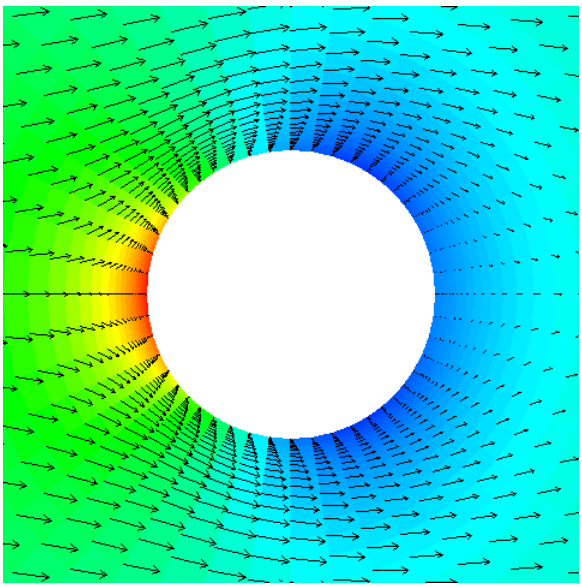

Figure 7: Color contours of pressure overlaid with **relative** velocity vectors for advection option #1.

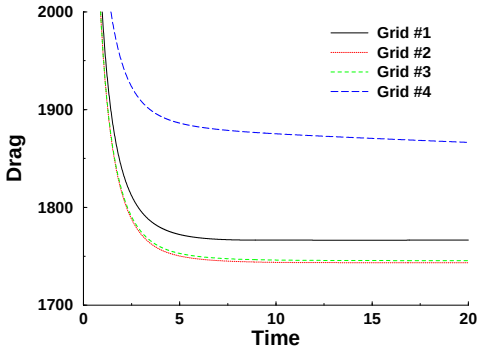

Figure 8: Time-history plot of drag for advection option #1 simulation.

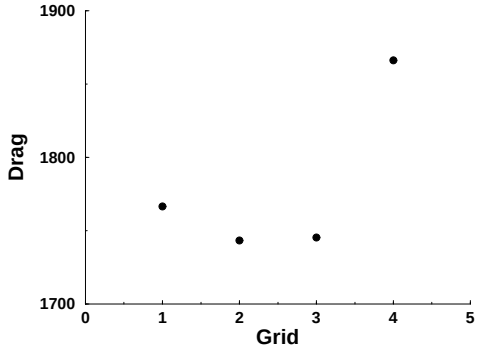

Figure 9: Convergence plot of drag for advection option #1 simulation.

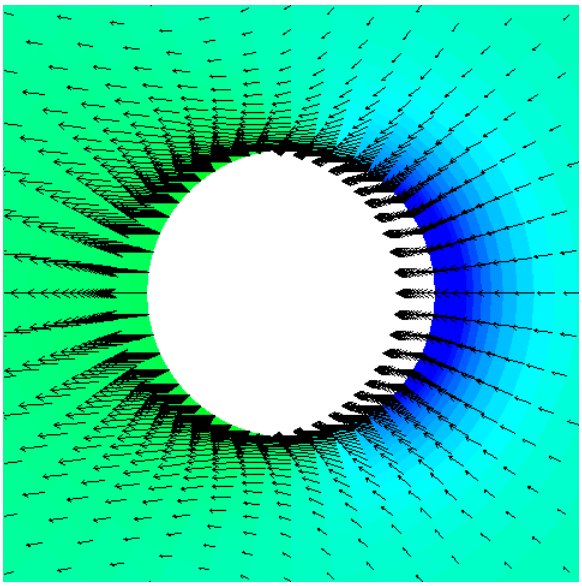

Figure 10: Color contours of pressure overlaid with velocity vectors for advection option #2.

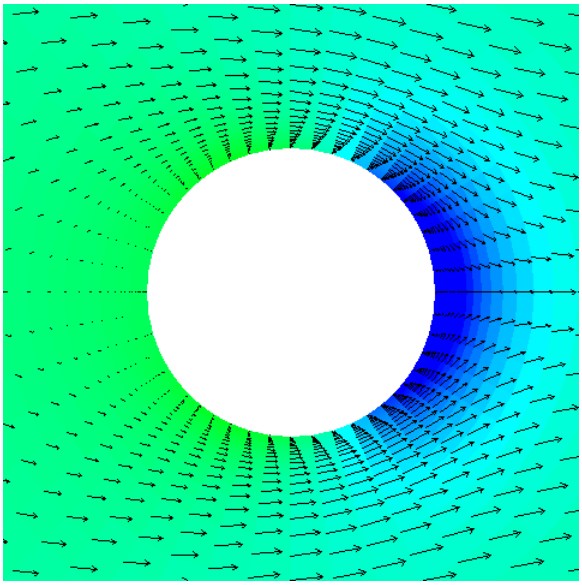

Figure 11: Color contours of pressure overlaid with **relative** velocity vectors for advection option #2.

expected to be realistic. Thus, these calculations should be rerun using a smaller time-step. This time-step restriction is the primary reason why Option #1 was implemented and why this was chosen as the topic of this project. The drag plots are presented in Figure 12 and Figure 13 for completeness.

Figure 14 summarizes this test case by plotting the drag time-history curves for grid 3 for the three cases. As expected from the earlier discussion, the drag value obtained in the fixed grid case agrees fairly well with the results from Advection Option #1. However, Advection Option #2 gives a much lower value for the drag.

5.2 Test Case #2: Buoyant Solid Sphere

The second test problem was designed to test the fluid-solid coupling in the code. It used Advection Option #1 for the grid motion. The test consisted of a hollow sphere made of a solid material immersed in an incompressible fluid. A gravitational body force was applied to the system, resulting in motion of the ball. The solid used a Gruneisen equation of state and a constitutive model with a

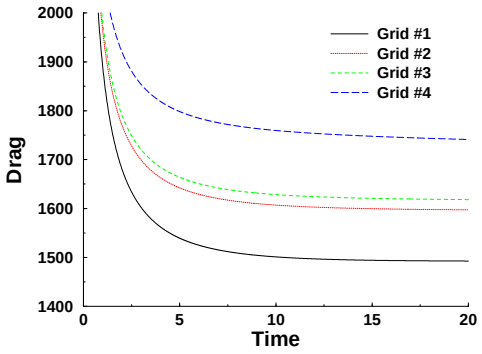

Figure 12: Time-history plot of drag for advection option #2 simulation.

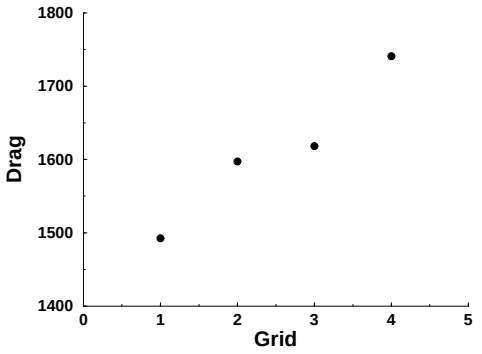

Figure 13: Convergence plot of drag for advection option #2 simulation.

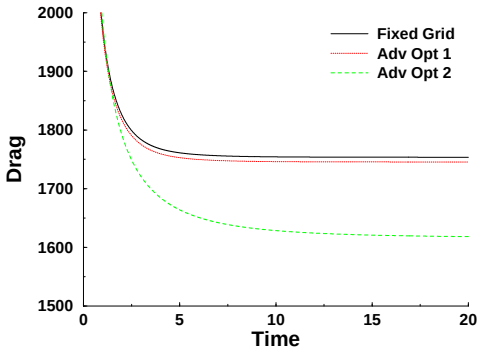

Figure 14: Time-history plot of drag for grid #3 for all cases.

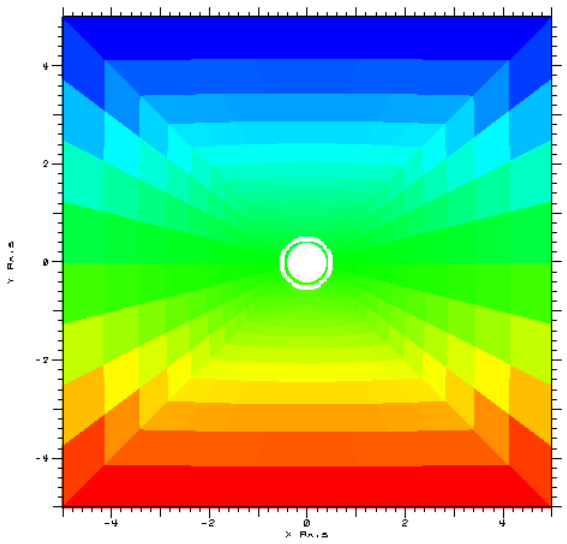

Figure 15: Initial Configuration and Hydrostatic Pressure

constant shear modulus. The following parameters were used:

$D_o = 1$	Outer Diameter of Sphere
$D_i = 0.8$	Inner Diameter of Sphere
$V_{init} = -0.1$	Initial Velocity of Sphere
$U_{\inf} = 0$	Initial Fluid Velocity
$\nu = 0.1$	Kinematic Viscosity
$\rho_f = 1000$	Fluid Density
$\rho_s = 1000$	Solid Density
$g = -10$	Gravitational Acceleration
$\gamma_0 = 2$	Gruneisen Gamma
$c = 1E6$	Solid Speed of Sound
$cmu = 1E15$	Shear Modulus of Solid

Note that this corresponds to a *very* stiff solid material and very little deformation should occur. Also notice that the fluid density is the same as the density of the solid. Therefore, since the ball is hollow, it should float. Since the sphere is initially given a small velocity downward, the buoyant forces will overtake the initial motion and the ball will change direction as the hydrostatic pressure pushes it upward.

The grid used in this simulation contained 768 solid elements and 9600 fluid elements. Only one grid was used due to the very large computational expense required to solve the problem. This problem ran 2500 time-steps and required approximately 30 seconds per cycle. Thus, approximately 20 hours of computer time were required. Over 90% of the time was spent computing the solid material.

Figure 15 shows the initial configuration of the problem. The solid white line in the middle represents the fluid-solid material boundary. The color contours

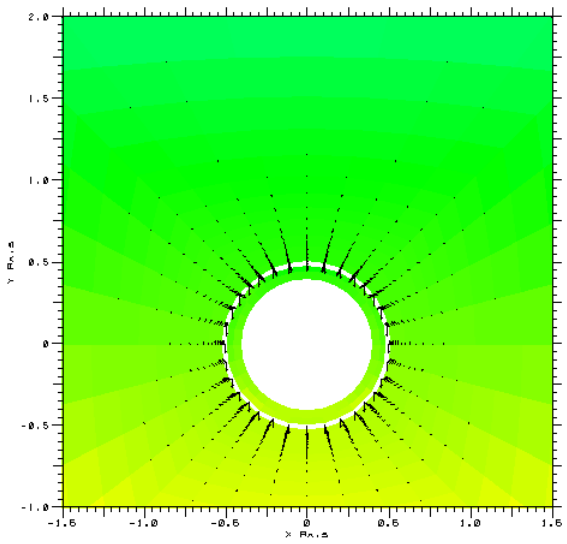

Figure 16: Initial Configuration and Flow Around Solid Ball

represent pressure. Notice that the fluid has established the hydrostatic pressure throughout the solution. Figure 16 shows the same configuration, but zoomed in to view the ball. Fluid velocity vectors are also added to the plot. Notice that the velocity vectors are initially pointing downward, following the motion of the ball. Figure 17 shows the final configuration of the simulation. At this point the ball has an upward velocity and has moved a distance a little more than the ball diameter. The motion of the solid has started to cause the fluid to recirculate. The fluid near the solid boundary is moving upward with the ball. Away from the sphere, the fluid is rushing downward to fill in the gap left by the moving solid.

The next plots provide a time-history of the motion of the ball. Figure 18 shows the vertical velocity of the sphere verses time. The sphere initially has a negative velocity, but as the hydrostatic pressure in the fluid pushes the ball upward the velocity increases. Towards the end of the simulation, drag begins to decelerate the sphere. It is unclear why the ball overshoots its terminal velocity. This will need to be investigated with a grid convergence study. Figure 19 shows the ball's position verses time (Figure 20 provides a zoomed in look at the start-up). As discussed for the velocity plot, the ball initially moves downward, then changes direction and moves upward.

Without a formal validation and grid convergence study no conclusions can be drawn to the accuracy of the method. However, the simulation does provide results that are qualitatively similar to what would be expected.

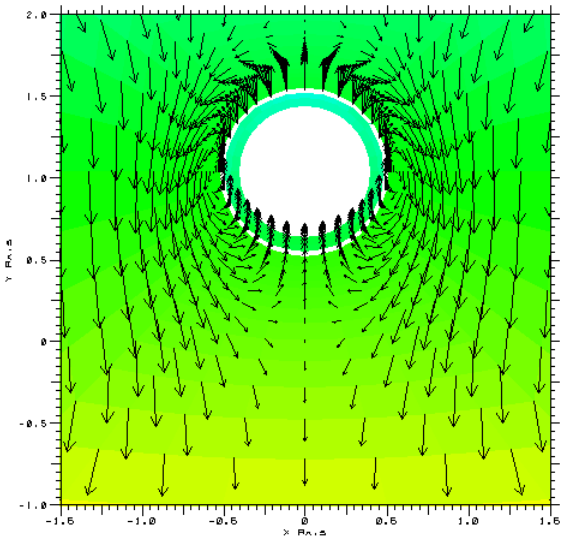

Figure 17: Final Configuration and Flow Around Solid Ball

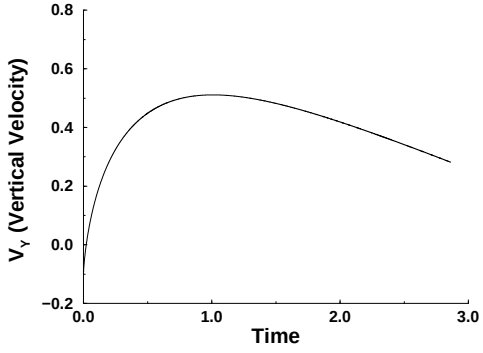

Figure 18: Vertical Velocity of Solid Ball

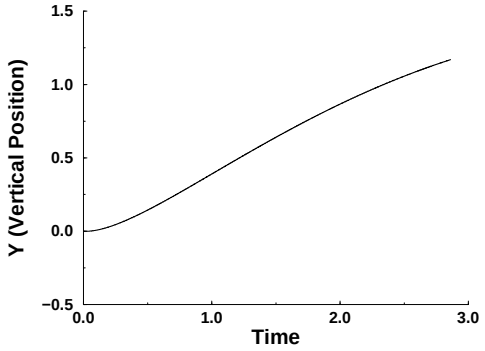

Figure 19: Vertical Position of Solid Ball

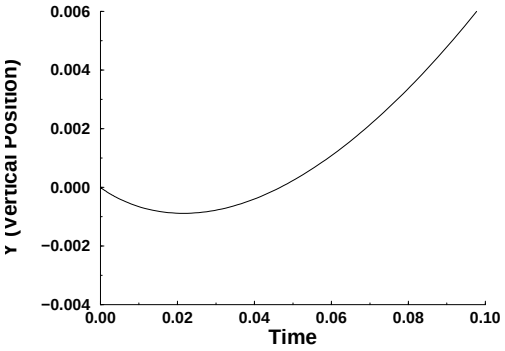

Figure 20: Vertical Position of Solid Ball

6 Conclusions

An incompressible flow code was coupled to an implicit solids code to investigate fluid-structure interaction. In order to do this, the fluids code was modified to allow grid motion using the ALE methodology. Two test cases were run: one to validate the grid motion capability in the fluid code and a second to check the fluid-structure coupling. A few conclusions are summarized here.

- Advection Option #1 was able to reasonably reproduce a fixed grid solution using a moving grid. Both agreed qualitatively with flow-field visualization experiments.
- Advection Option #2 did not predict the correct flow. It is assumed that this was caused by time-steps which were too large. This should be investigated more.
- The implicit projection method used for the fluid equations requires "reasonably" sized time-steps for accurate solutions. This could have influenced the convergence study results.
- Test problem 1 should be investigated further with an analysis of time-discretization effects.
- The code was able to provide some insight into the fluid-structure interaction for a buoyant sphere.
- A more detailed grid-resolution study is required for test problem 2.

The following items may be considered for future projects to improve the performance of the code:

- Non-linear iteration may be required for convergence between the fluid and solid.
- Strong coupling may be required–solve both fluid and solid in the same matrix solution.
- Iteration within the incompressible flow step to better address the non-linear advective term could improve the projection algorithm.

This is an ongoing project. There is still plenty of work to be done.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

References

[1] Uchiyama T, ALE finite element method for gas-liquid two-phase flow including moving boundary based on an incompressible two-fluid model. *Nuclear Engineering and Design*, 2001; **205**:69–82.

[2] Benson DJ, Computational methods in Lagrangian and Eulerian hydrocodes. *Computer Methods in Applied Mechanics and Engineering*, 1992; **99**:235–394.

[3] Comini G, Del Giudice S, Nonino C, *Finite Element Analysis in Heat Transfer Basic Formulation and Linear Problems*, 1994, Taylor & Francis.

[4] Gresho PM, Chan ST, Lee RL, Upson CD. A modified finite element method for solving the time-dependent, incompressible Navier-Stokes equations part 1: theory. *International Journal for Numerical Methods in Fluids* 1984; **4**(6):557–598.

[5] Goudreau GL, Hallquist JO. Recent developments in large-scale finite element Lagrangian hydrocode technology. *Computational Methods in Applied Mechanics and Engineering* 1982; **33**(1-3):725-757.

[6] Gresho PM, Chan ST, Projection 2 goes turbulent–and fully implicit. *International Journal of Computational Fluid Dynamics*, 1998; **9**(3-4):249–272.

[7] Silvester DJ, Kechkar N, Stabilised bilinear-constant velocity-pressure finite elements for the conjugate gradient solution of the Stokes problem. *Computer Methods in Applied Mechanics and Engineering*, 1990; **79**:71–86.

[8] Becker R, The implicithydro package in ALE3D. Unpublished Draft, 2004.

[9] Van Dyke M, *An Album of Fluid Motion*, 1982, The Parabolic Press.